# RSA with Balanced Short Exponents and Its Application to Entity Authentication

Hung-Min Sun[1] and Cheng-Ta Yang[2]

[1] Department of Computer Science,
National Tsing Hua University, Hsinchu, Taiwan 30055
hmsun@cs.nthu.edu.tw

[2] Department of Computer Science and Information Engineering,
National Cheng Kung University

**Abstract.** In typical RSA, it is impossible to create a key pair $(e, d)$ such that both are simultaneously much shorter than $\phi(N)$. This is because if $d$ is selected first, then $e$ will be of the same order of magnitude as $\phi(N)$, and vice versa. At Asiacrypt'99, Sun et al. designed three variants of RSA using prime factors $p$ and $q$ of unbalanced size. The first RSA variant is an attempt to make the private exponent $d$ short below $N^{0.25}$ and $N^{0.292}$ which are the lower bounds of $d$ for a secure RSA as argued first by Wiener and then by Boneh and Durfee. The second RSA variant is constructed in such a way that both $d$ and $e$ have the same bit-length $\frac{1}{2}\log_2 N + 56$. The third RSA variant is constructed by such a method that allows a trade-off between the lengths of $d$ and $e$. Unfortunately, at Asiacrypt'2000, Durfee and Nguyen broke the illustrated instances of the first RSA variant and the third RSA variant by solving small roots to trivariate modular polynomial equations. Moreover, they showed that the instances generated by these three RSA variants with unbalanced $p$ and $q$ in fact become more insecure than those instances, having the same sizes of exponents as the former, in RSA with balanced $p$ and $q$. In this paper, we focus on designing a new RSA variant with balanced $d$ and $e$, and balanced $p$ and $q$ in order to make such an RSA variant more secure. Moreover, we also extend this variant to another RSA variant in which allows a trade-off between the lengths of $d$ and $e$. Based on our RSA variants, an application to entity authentication for defending the stolen-secret attack is presented.

**Keywords:** RSA, Short Exponent Attack, Lattice Reduction, Entity Authentication

## 1 Introduction

RSA [14], the most popular public key cryptosystem, was announced in 1978 by Rivest, Shamir, and Adleman at MIT. However, RSA suffers from heavy computation because it requires exponentiation operations modulo a large integer $N$ ($N = pq$, a product of two large primes). The RSA encryption and decryption time is almost proportional to the number of bits in the exponent. In order to

© International Association for Cryptologic Research 2005

reduce the RSA encryption (signature verification) time or decryption (signature generation) time, it is important to choose a small public exponent or a short private exponent. Generally speaking, in standard RSA, encryption are much faster than decryption because the public exponent is usually selected as $2^{16}+1$, while the private exponent is of the same order of magnitude as $\phi(N)$. In some applications, one would like to accelerate decryption process. Thus selecting a short private exponent is preferred. In such a case, the encryption will be cost-inefficient because the size of public exponent will be of the same order of magnitude as $\phi(N)$. Towards to the use of RSA with short private exponent, one must be careful with the short exponent attacks on RSA. In 1990, Wiener [21] first showed that the instances of RSA cryptosystem with short secret exponent ($d < N^{0.25}$) are insecure because one could find the short private exponent $d$ in polynomial time by using the continued fractions algorithm. In 1999, Boneh and Durfee [2] showed how to improve the bound of Wiener up to $d < N^{0.292}$. Their attack is based on the famous $L^3$-lattice reduction algorithm [10] by Coppersmith [4] on finding small roots of particular bivariate modular polynomial equations.

At Asiacrypt'99, Sun, Yang, and Laih [17, 18] designed three variants of RSA using prime factors $p$ and $q$ of unbalanced size. The first RSA variant is an attempt to make the private exponent $d$ short below Wiener's bound [21] and Boneh and Durfee's bound [2]. In this variant, the RSA system is constructed from $p$ and $q$ of different sizes in order to defend against the well-known short private exponent attacks. They claimed that when $p$ and $q$ are unbalanced enough, $d$ can be even smaller than $N^{0.25}$. A suggested choice of parameters is: $p$ of 256 bits, $q$ of 768 bits, and $d$ of 192 bits. Note that in this variant, $e$ is determined as that in typical RSA, hence $e$ is of 1024 bits. The second RSA variant is constructed in such a way that both $d$ and $e$ have the same bit-length $\frac{1}{2}\log_2 N + 56$ by choosing unbalanced $p$ of $\frac{1}{2}\log_2 N - 112$ bits and $q$ of $\frac{1}{2}\log_2 N + 112$ bits respectively. The motivation of this variant is for balancing and minimizing both public and private exponents. A suggested choice of parameters is: $p$ of 400 bits, $q$ of 624 bits, $d$ of 568 bits, and $e$ of 568 bits. The third RSA variant is constructed by such a method that allows a trade-off between the lengths of $d$ and $e$ (that is $\log_2 e + \log_2 d \approx \log_2 N + l_k$, where $l_k$ is a predetermined constant, e.g., 112) under the limitation of $\log_2 p + \log_2 d \leq \log_2 N$ (assuming $p < q$). The purpose of this variant is for rebalancing the computation cost between encryption and decryption. By this method, one may shift the work from decryptor to encryptor. An illustrated instance of RSA has the parameters: $p$ of 256 bits, $q$ of 768 bits, $d$ of 256 bits, and $e$ of 880 bits. Unfortunately, Durfee and Nguyen [5] broke the illustrated instances of the first RSA variant and the third RSA variant by solving small roots to trivariate modular polynomial equations. They also showed that the instances generated by these three RSA variants with unbalanced $p$ and $q$ in fact become more insecure than those instances, having the same sizes of exponents as the former, in RSA with balanced $p$ and $q$. In this paper, we are interested in enhancing the security of Sun et al.'s RSA variants by using balanced $p$ and $q$. It is clear that for the first RSA variant, the improved one with balanced $p$ and $q$ is in fact the standard RSA. Hence, it is impossible to make $d$ short below Boneh and Durfee's bound and Wiener's bound. Therefore, we will not focus on the first variant. For the second RSA variant, it is unable to make $p$ and $q$ balanced because $p$ is of $\frac{1}{2}\log_2 N - 112$ bits and $q$ is of $\frac{1}{2}\log_2 N + 112$ bits in this variant. For the third RSA variant, the possible constructed RSA with balanced $p$ and $q$ are only those instances of RSA with $d$ of $\frac{1}{2}\log_2 N$ bits and $e$ of $\frac{1}{2}\log_2 N + l_k$ bits. This is due to the limitation of $\log_2 p + \log_2 d \leq \log_2 N$. In this paper, we focus on designing a new RSA variant with balanced $p$ and $q$, and balanced $d$ and $e$ in order to make such an RSA variant more secure against the Durfee-Nguyen attack and the other existing attacks. Moreover, we also extend our variant to another RSA variant in which $p$ and $q$ are balanced and $\log_2 e + \log_2 d \approx \log_2 N + l_k$. Compared with RSA using CRT-based decryption (RSA-CRT for short), our schemes seem not to provide better performance for decryption. However it is still an interesting topic like those short exponent attacks [2, 21] working on the standard RSA. Moreover, based on our schemes, we present an application to entity authentication for defending the stolen-secret attack. On the contrary, RSA-CRT can not be applied to the application. We refer the readers to Section 7.

This paper is organized as follows. In Section 2, we review the standard RSA, RSA-CRT, Sun et al.'s RSA variants, and recall some well-known attacks on RSA with short private exponent. In Section 3, we present a new RSA variant with balanced $p$ and $q$, and balanced $e$ and $d$; and show the flexibility for constructing such an RSA variant. In Section 4, we analyze the security of this proposed RSA variant. In Section 5, we extend the proposed RSA variant in Section 3 to another RSA variant in which $p$ and $q$ are balanced and $\log_2 e + \log_2 d \approx \log_2 N + l_k$. In Section 6, we show the experimental results of our implementations for our schemes. In Section 7, we compare our RSA variants with RSA-CRT, and give an application based on our RSA variants. Finally, we conclude this paper in Section 8.

## 2 Preliminaries

### 2.1 Description of Notations

The notations in Table 1 are used throughout this paper.

### 2.2 The Standard RSA and RSA-CRT

In standard RSA, $N = p \times q$ is the product of two large primes $p$ and $q$. The public exponent $e$ and private exponent $d$ satisfy $e \times d \equiv 1 \bmod \phi(N)$, where $\phi(N) = (p-1)(q-1)$ is the Euler totient function of $N$. Here, $N$ is called the RSA modulus. The public key is the pair $(N, e)$ that is used for encryption (or signature verification): $c = m^e \bmod N$. The private key $d$ is to enable decryption of ciphertext (or signature generation): $m = c^d \bmod N$. Traditionally, we select two primes (of 512 bits) $p$ and $q$ first, and then multiply them to obtain $N$ (about 1024 bits). Next, we pick the public exponent $e$ first, and then determine the

**Table 1.** Notations.

| | |
|---|---|
| $p, q$ : | The two large primes of RSA. |
| $N$ : | The product of two large prime factors $p$ and $q$, i.e. $N = p \times q$. |
| $e, d$ : | The public exponent and private exponent, $ed \equiv 1 \bmod \phi(N)$. |
| $\Delta$ : | The prime difference, $\Delta = \|p - q\|$. |
| $\delta$ : | $d = N^{\delta}$. |
| $\varpi$ : | $e = N^{\varpi}$. |
| $\gamma$ : | $\|p - q\| = N^{\gamma}$. |
| $l_X$ : | The bit-length of a variable $X$. |

private exponent $d$ by $d \equiv e^{-1} \bmod \phi(N)$, or we select the private exponent $d$ first, and then compute the public exponent $e$ by $e \equiv d^{-1} \bmod \phi(N)$. For the deduction mentioned above, either $e$ or $d$ is of the same order of magnitude as $\phi(N)$. Instead of computing $m = c^d \bmod N$, RSA-CRT computes $m_1 = c^{d_p} \bmod p$, and $m_2 = c^{d_q} \bmod q$, where $d_p = d \bmod p - 1$ and $d_q = d \bmod q - 1$, then applying the Chinese Remainder Theorem, one may easily recover $m$ by $m_1$ and $m_2$.

## 2.3 Sun, Yang, and Laih's RSA Variants

At Asiacrypt'99, Sun et al. [17, 18] designed three variants of RSA using prime factors $p$ and $q$ of unbalanced size. The first variant of RSA is an attempt to make the private exponent $d$ short below Wiener's bound and Durfee and Nguyen's bound. In this variant, the RSA system is designed by unbalanced $p$ and $q$ in order to defend against all existing attacks on short private exponent. The second variant of RSA is an attempt to balance and minimize both public and private exponents. It is constructed in such a way that both $d$ and $e$ have the same size of $\frac{1}{2}\log_2 N + 56$ bits by choosing unbalanced $p$ of $\frac{1}{2}\log_2 N - 112$ bits and $q$ of $\frac{1}{2}\log_2 N + 112$ bits respectively. The third variant of RSA is an attempt to rebalance the computation cost between encryption and decryption. By this variant, one may shift the work from decryptor to encryptor. It is constructed by such a method that allows a trade-off between the lengths of $d$ and $e$ (that is $\log_2 e + \log_2 d \approx \log_2 N + 112$) under the limitation of $\log_2 p + \log_2 d \leq \log_2 N$. Due to the limit of space, we describe the details of these three RSA variants in Appendix A.

Very soon, Durfee and Nguyen [5] broke the illustrated instances of the first RSA variant and the third RSA variant. Moreover, they showed that the instances generated by these three RSA variants with unbalanced $p$ and $q$ in fact become more insecure than those instances, having the same sizes of exponents as the former, in RSA with balanced $p$ and $q$. We describe their attack later.

## 2.4 Attacks on RSA with Short Private Exponent

**Wiener's Attack and Its Extensions.** Wiener's attack [21] is based on continued fractions algorithm to find the numerator and denominator of a fraction in polynomial time when a sufficiently close estimate of the fraction is known. He showed that the RSA system can be totally broken if the private exponent is up to approximately one-quarter as many bits as the modulus under both $p$ and $q$ of approximately the same size. For simplicity, we slightly modify Wiener's attack in the following way. Since $ed \equiv 1 \bmod \phi(N)$, there exists a $k$, $\gcd(d, k) = 1$, such that $ed = k\phi(N) + 1$. So, $|\frac{e}{\phi(N)} - \frac{k}{d}| = \frac{1}{d\phi(N)}$. Hence, $\frac{k}{d}$ is an approximation of $\frac{e}{\phi(N)}$. We can rewrite the equation: $ed = k\phi(N)+1$ as: $ed = k(N-(p+q)+1)+1$. As pointed out by Pinch [12], if $p < q < 2p$ and $d < \frac{1}{3}N^{0.25}$, then $p+q-1 < 3\sqrt{N}$ and $k < d < \frac{1}{3}N^{0.25}$. Using $N$ in place of $\phi(N)$, we obtain:

$$|\frac{e}{N} - \frac{k}{d}| = \frac{k(p+q-1-\frac{1}{k})}{dN} \leq \frac{1}{dN^{0.25}} < \frac{1}{3d^2} < \frac{1}{2d^2}.$$

Thus $\frac{k}{d}$can be found because it is one of the $\log N$ convergents of the continued fraction for $\frac{e}{N}$.

The extension of Wiener's attack was proposed by Verheul and Tilborg [19]. When $d > N^{0.25}$, their attack needs to do an exhaustive search for about $2t + 8$ bits, where $t \approx \log_2(\frac{d}{N^{0.25}})$. In addition, Weger [20] further proposed another extension of Wiener's attack in the case when the prime difference of $N$, $\Delta = |p - q|$, is small. Let the prime difference $\Delta = |p - q| = N^{\gamma}$ for $0.25 \leq \gamma \leq 0.5$, and $d = N^{\delta}$. Weger showed that if $\delta < \frac{3}{4} - \gamma$, one could find the short private exponent $d$ using Wiener's attack. Thus Weger improved Wiener's bound from $\delta < 0.25$ to $\delta < \frac{3}{4} - \gamma$.

**The Boneh-Durfee Attack and Its Extension.** Based on solving the small inverse problem, Boneh and Durfee [2] proposed another attack on RSA with short private exponent, which leads to a better bound than that proposed by Wiener [21]. They concluded that if $e \approx N$ and $d < N^{0.292}$, then the private exponent $d$ can be found efficiently.

In typical RSA system, $ed = k\phi(N) + 1$, $e = N^{\varpi}$ and $d = N^{\delta}$. So, $ed = k(p-1)(q-1)+1 = k((N+1)-(p+q))+1$. Let $A = N + 1$, $s = -(p+q)$, and $t = -k$. Then $ed + t(A + s) = 1$. Thus, $t(A + s) \equiv 1 (\bmod\ e)$ and we can bound $s$ and $t$ by $|t| < 3e^{1+\frac{\delta-1}{\varpi}}$ and $|s| < 2e^{\frac{1}{(2\varpi)}}$. Boneh and Durfee took $\varpi \approx 1$ and ignored small constants, and ended up with the following problem: finding integer $t$ and $s$ such that $t(A + s) \equiv 1(\bmod\ e)$ where $|s| < e^{0.5}$ and $|t| < e^{\delta}$.

Now, we have a simple review of the lattice theory first. Let $v_1,..., v_w \in Z^n$ be linearly independent vectors with $w \leq n$. A lattice $L$ spanned by $\langle v_1, ..., v_w\rangle$ is the set of all integer combinations of $v_1,..., v_w$. We denote by $v_1^*,..., v_w^*$ the vectors obtained by applying the Gram-Scmidt process to the vectors $v_1,..., v_w$. We define the determinant of the lattice $L$ as $det(L) := \prod_{i=1}^{w} ||v_i^*||$, where $||.||$ denotes the Euclidean norm on vectors. We say that the lattice is full rank if $w = n$. For a lattice $L$ spanned by $\langle v_1, ..., v_w\rangle$, the LLL algorithm runs in polynomial time and produces a new basis $\langle r_1, ..., r_w\rangle$ of $L$ as $||r_1|| \leq 2^{\frac{w}{2}} det(L)^{\frac{1}{w}}$ and $||r_2|| \leq 2^{\frac{(w-1)}{2}} det(L)^{\frac{1}{(w-1)}}$, $r_1$ and $r_2$ are two shortest vectors in the new basis.

Boneh and Durfee solved the small inverse problem by using Coppersmith's approach [4]. Recall that let $h(x, y) \in Z[x, y]$ be a polynomial which is a sum of at most $w$ monomials. Suppose that (1) $h(x_0, y_0) \equiv 0 \bmod e^m$ for some positive integer $m$ where $|x_0| < X$ and $|y_0| < Y$ , and (2) $||h(xX, yY)|| < e^m/\sqrt{w}$, then $h(x_0, y_0) = 0$ holds over the integers.

The small inverse problem is the following: given a polynomial $f(x, y) = x(A + y) - 1$, find an $(x_0, y_0)$ as $f(x_0, y_0) \equiv 0 (\bmod\ e)$ where $|x_0| < e^\delta$ and $|y_0| < e^{0.5}$. We would find a polynomial with a small norm that has $(x_0, y_0)$ as a root modulo $e^m$ for some positive integer $m$. Boneh and Durfee defined the polynomials $g_{i,k}(x, y) = x^i f^k(x, y)e^{m-k}$ and $h_{j,k}(x, y) = y^j f^k(x, y)e^{m-k}$,for $k = 0, ..., m$, where $g_{i,k}(x, y)$ is called x-shifts and $h_{j,k}(x, y)$ is called y-shifts. For each $k$, they used $g_{i,k}(xX, yY)$ for $i = 0, ..., m - k$ and used $h_{j,k}(xX, yY)$ for $j = 0, ..., t$, where $t$ is minimized based on $m$. Observe that the matrix is triangular and has a dimension $\frac{(m+1)(m+2)}{2} + t(m + 1)$. The determinant of the lattice can be easily computed as the product of the diagonal entries

$$det_x = e^{m(m+1)(m+2)/3} \cdot X^{m(m+1)(m+2)/3} \cdot Y^{m(m+1)(m+2)/6}$$
$$det_y = e^{tm(m+1)/2} \cdot X^{tm(m+1)/2} \cdot Y^{t(m+1)(m+t+1)/2}.$$

Let $det(L) = det_x \cdot det_y$. By Ignoring the denominator in order to simplify the derivations, we get the condition $det(L) < e^{mw}$. Finally, on the basis of the lattice theory and Coppersmith's approach, We deduce that

$$\delta < \frac{7}{6} - \frac{1}{3}\sqrt{7 + \frac{16}{m} + \frac{4}{m^2}} + \frac{5}{6m}.$$

For large $m$, this converges to $\delta < \frac{7}{6} - \frac{\sqrt{7}}{3} \approx 0.285$. By working on a sub-lattice, the bound on $\delta$ can be improved to $\delta < 1 - \frac{\sqrt{2}}{2} \approx 0.292$.

Another improvement was proposed by Wager [20]. He showed that RSA is insecure when the length $\delta$ of the private exponent is in $2-4\gamma < \delta < 1-\sqrt{2\gamma - \frac{1}{2}}$, where $|p - q| = N^\gamma$ and $d = N^\delta$.

**The Cubic Attack.** Here, we review the cubic attack in [17, 18]. In RSA, $N = pq$ and $ed = k(p - 1)(q - 1) + 1$, therefore, the modular equations are $k(p - 1)(q - 1) + 1 \equiv 0(\bmod\ e)$ and $pq \equiv N(\bmod\ e)$. According to the above two equations, we can obtain one cubic equation with two variables $k$ and $p$ : $k(p - 1)(N - p) + p \equiv 0(\bmod\ e)$. If $\log_2 k + \log_2 p < \frac{1}{3} \log_2 e$, we can solve such a cubic equation heuristically using Coppersmith's technique [4].

## 2.5 The Durfee-Nguyen Attack and Its Extension

Extending the Boneh-Durfee attack, Durfee and Nguyen [5] attacked Sun *et al.'s* RSA variants by solving small roots to trivariate modular polynomial equations using Coppersmith's lattice technique. From the RSA equation $ed = k\phi(N)+1 = k(p - 1)(q - 1) + 1$, let $A = N + 1$, it implies $1 + k(A - p - q) \equiv 0(\bmod\ e)$.

**Table 2.** Largest $\delta$ (where $d < N^{\delta}$) for which Durfee-Nguyen's attack can be completed.

| | | $\log_N(e)$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1.0 | 0.9 | 0.86 | 0.8 | 0.7 | 0.6 | 0.55 |
| | 0.5 | 0.284 | 0.323 | 0.339 | 0.363 | 0.406 | 0.451 | 0.475 |
| | 0.4 | 0.296 | 0.334 | 0.350 | 0.374 | 0.415 | 0.460 | $0.483_{\mathrm{II}}$ |
| $\log_N(p)$ | 0.3 | 0.334 | 0.369 | 0.384 | 0.406 | 0.446 | 0.487 | 0.510 |
| | 0.25 | $0.364_{\mathrm{I}}$ | 0.398 | $0.412_{\mathrm{III}}$ | 0.433 | 0.471 | 0.511 | 0.532 |
| | 0.2 | 0.406 | 0.437 | 0.450 | 0.470 | 0.505 | 0.542 | 0.562 |
| | 0.1 | 0.539 | 0.563 | 0.573 | 0.588 | 0.615 | 0.644 | 0.659 |

They treated the above equation as a trivariate equation modular $e$ with three unknown variables, $k$, $p$, $q$, with the special property that the product $pq$ of two of them is the known quantity $N$. Here, the problem is regarded as given a polynomial $f(x, y, z) = x(A + y + z) - 1$, finding an integer solution $(x_0, y_0, z_0)$ satisfying the equation $f(x_0, y_0, z_0) \equiv 0 (\bmod\ e)$ where $|x_0| < X$, $|y_0| < Y$, $|z_0| < Z$, and $y_0 z_0 = N$. Note that the bounds are $X \approx \frac{ed}{N}$, $Y \approx p$, and $Z \approx q$.

To search for low-norm integer linear combinations of these polynomials of the form $e^{m-v} x^{u1} y^{u2} z^{u3} \cdot f^v(x, y, z)$, they chose the polynomials $g_{k,i,b}(x, y, z) := e^{m-k} x^i y^a z^b f^k(x, y, z)$, for $k = 0..(m-1)$, $i = 1..(m-k)$,and $b = 0, 1$; and, $h_{k,j}(x, y, z) := e^{m-k} y^{a+j} f^k(x, y, z)$, for $k = 0..m$ and $j = 0..t$, then fixed an integer $m$, and let $a$ and $t > 0$ be integers which would be optimized later. Following the LLL algorithm [4], they obtained two short vectors corresponding to polynomials $h_1(x, y, z)$, $h_2(x, y, z)$ that had $(k, p, q)$ as a root over the integers; and letting $z = \frac{y}{N}$, they deduced these polynomials to bivariate polynomials $H_1(x, y)$ and $H_2(x, y)$ which had $(k, p)$ as a solution. Taking the resultant $Res_x(H_1(x, y), H_2(x, y))$ produced an univariate polynomial $H(y)$ which had $p$ as a root. They summarized the largest possible $\delta$ for which their attack could succeed as shown in Table 2.

From Table 2, we conclude that instances from RSA with unbalanced $p$ and $q$ are in fact more insecure than those from RSA with unbalanced $p$ and $q$.An improvement of Durfee-Nguyen's largest $\delta$ was proposed by Hong *et al. in* [7]. They showed how to improve the bound from 0.483 to 0.486 when $\log_N(p) \approx 0.4$, $\log_N(e) \approx 0.55$ using Coppersmith's theorem [4]. Because their attack is very similar to the Durfee-Nguyen attack, we omit to review the details of their attack.

## 3 New RSA Variant with Balanced Exponents and Balanced Prime Factors

Sun et al.'s second variant is designed for balancing and minimizing both public and private exponents. An illustrated instance of this variant was given in [17, 18]. The illustrated instance has parameters: $p$ of 400 bits, $q$ of 624 bits, $d$ of 568 bits, and $e$ of 568 bits. Although this instance is still secure against the Durfee-Nguyen attack, however, as shown in Table 2, an instance of RSA with the same size of $d$ and $e$, and balanced $p$ and $q$ is more secure than the illustrated instance in [17, 18]. Unfortunately, it is impossible to make $p$ and $q$ balanced in Sun et al.'s second variant because $p$ is of $\frac{1}{2}\log_2 N - 112$ bits and $q$ is of $\frac{1}{2}\log_2 N + 112$ bits. In this section, we present a new RSA variant in which $d$ and $e$ are balanced, and $p$ and $q$ are also balanced.

## 3.1 The Proposed Scheme

Our scheme is based on the Extended Euclidean algorithm [6]. Recall that for two integers $a, b > 1$, if $\gcd(a, b) = 1$, then we can find a unique pair $(u_h, v_h)$ satisfying $au_h - bv_h = 1$,where $(h-1)b < u_h < hb$ and $(h-1)a < v_h < ha$, for any integer $h \geq 1$. Our method is as follows:

**Scheme A:** input: $l_N$ and $w$; output: $e, d, p, q$ and $N$.

```
Step 1. Randomly select a prime $p$ of $\frac{1}{2}l_N$ bits.
Step 2. Randomly select a number $k^{'}$, such that $k^{'}(p-1)$ is of $\frac{1}{2}l_N + w$ bits,
        where $w$ is a security parameter, e.g., $w = 56$.
Step 3. Randomly select a number $d$ of $\frac{1}{2}l_N + w$ bits, such that $\gcd(k^{'}(p-1), d) = 1$.
Step 4. Determine $u^{'}, v^{'}$ such that $du^{'} - k^{'}(p-1)v^{'} = 1$, where $0 < u^{'} < k^{'}(p-1)$ and $0 < v^{'} < d$.
Step 5. If $l_{v'} < \frac{1}{2}l_N + w$, then assign $u^{'} = u^{'} + k^{'}(p-1)$ and $v^{'} = v^{'} + d$.
Step 6. Try to find $v^{'} = k^{''}q^{'}$, where $l_{k''} = w$ and $q^{'} + 1$ is a prime. If this
        fails, go to Step 3.
Step 7. Let $e = u^{'}$, $q = q^{'} + 1$, and $N = pq$.
```

The algorithm will generate RSA instances in which both $p$ and $q$ are approximately $\frac{1}{2}\log_2 N$ bits long, and both $e$ and $d$ are approximately $(\frac{1}{2}\log_2 N + w)$ bits long. Also the resulting $e$ and $d$ will satisfy $ed = k^{'}k^{''}(p-1)(q-1)+1 = k\phi(N)+1$, where $k = k^{'}k^{''}$. Note that the prime $p$ generated in Step 1 can be determined arbitrarily, e.g., by selecting a strong prime $p$, but the prime $q$ generated in Step 7 cannot. Fortunately, for an RSA key the requirement that $p$ and $q$ are strong primes is no longer needed due to [15]. As an example, we construct an instance of RSA that $p$ is of 512 bits, $q$ is of 513 bits, and $e$ and $d$ are 568 bits (assigning $\log_2 N \approx 1024$, and $w = 56$). We show this instance in Appendix B.

## 3.2 Feasibility for the Algorithm

In this section, we show that the proposed algorithm in Section 3.1 is feasible. Without loss of generality, we assume $\log_2 N \approx 1024$, and $w = 56$. The critical step in the above algorithm is Step 6. Because $v^{'}$ is about of 568 or 569 bits, we will try to find a lower bound for the probability of that being given a random number $x$ of 568 or 569 bits, it can be expressed in the form $x = yz$ satisfying $l_x = 568$ or $569$, $l_y = 56$, $l_z = 512$ or $513$ or $514$, and $z + 1$ being a prime.

**Theorem 1.** *The probability that given a randomly selected number $x$ of 568 or 569 bits, it can be expressed in the form $x = yz$ satisfying $l_x = 568$ or 569, $l_y = 56$, $l_z = 512$ or 513 or 514, and $z + 1$ being a prime is much higher than $\frac{1}{387618}$.*

*Proof.* We omit the details due to the limit of space.

Based on Theorem 1, the existence and its probability for a random number which can go through Step 6 in the proposed scheme has been evaluated. Now we consider the cost for factoring a 568-bit $v^{'}$ into the form: $k^{''}q^{'}$, where $l_{k^{''}} = 56$, in Step 6. Given a number $v^{'}$, it is easy for us to find all prime factors of $v^{'}$ which are less than 56 bits by some well-known factoring algorithms, such as ECM algorithm [8]. Then we can try to combine these prime factors to form a 56-bit $k^{''}$ in polynomial time.

## 4 Security Considerations

In this section, we analyze our scheme to thwart the previous well-known attacks on short private exponent, including Wiener's attack [21], the Boneh-Durfee attack [2], the Durfee-Nguyen attack [5], the cubic attack [17, 18], and their extensions [7, 19, 20].

**Defending Against Wiener's Attack.** We will check the security of our RSA variant according to Wiener's attack. It is clear that

$$|\frac{e}{N} - \frac{k}{d}| = \frac{k}{d} \times \frac{p + q - 1 - \frac{1}{k}}{N} > \frac{k}{d}\frac{q}{N}.$$

In our variant, $p$ and $q$ are about of 512 bits, and $e$ and $d$ are about of 568 bits, so $2^{511} \leq p < 2^{512}$, $2^{111} \leq k < 2^{112}$, $2^{567} \leq e < 2^{568}$, $2^{567} \leq d < 2^{568}$. Now, we can obtain $|\frac{e}{N} - \frac{k}{d}| > \frac{k}{d}\frac{q}{N} > \frac{2^{111}}{2^{568}} \times \frac{2^{511}}{2^{1024}} = \frac{1}{2^{970}} \gg \frac{1}{2d^2} \approx \frac{1}{2^{1136}}$. Thus, Wiener's attack does not apply to our scheme.

**Defending Against the Boneh-Durfee Attack and the Durfee-Nguyen Attack.** Following Boneh and Durfee's approach, let $A = N + 1$, $s = -(p + q)$, and $t = -k$. Thus $t(A + s) \equiv 1 (\bmod\ e)$. Let $|s| < e^{\alpha}$ and $|t| < e^{\beta}$. The sufficient condition for solving the small inverse problem is: $4\alpha(2\beta + \alpha - 1) < 3(1 - \beta - \alpha)^2$.

In our example, $p$ and $q$ are about of 512 bits, and $e$ and $d$ are about of 568 bits, therefore, $2^{511} \leq p < 2^{512}$, $2^{111} \leq k < 2^{112}$, $2^{567} \leq e < 2^{568}$, $2^{511} \leq d < 2^{512}$. We can calculate $|s| = |p + q| = e^{\alpha}$, $|k| = e^{\beta}$, i.e. $2^{512} < (2^{568})^{\alpha}$, $2^{112} < (2^{568})^{\beta}$, we can get $\alpha \approx \frac{512}{568}$, $\beta \approx \frac{112}{568}$ respectively. It is clear that $4\alpha(2\beta + \alpha - 1) = 1.06645 \gg 0.02916 = 3(1 - \beta - \alpha)^2$. So, the Boneh-Durfee attack cannot succeed.

Next, we examine the largest $\delta$ (where $d < N^{\delta}$) for which the Durfee-Nguyen attack [5] can succeed. Our $p$ is of 512 bits, then $\log_N(p) \approx 0.5$; $e$ is of 568 bits, then $\log_N(e) \approx 0.55$; and $d$ is of 568 bits. So, we can figure out $d \approx N^{\frac{568}{1024}} \approx N^{0.55} > N^{0.475}$. So our RSA variant is secure against the Durfee-Nguyen attack.

Finally, we check the prime difference that Weger proposed.

$2 - 4\gamma = 2 - 4 \times \frac{1}{2} = 0$ and $1 - \sqrt{2\gamma - \frac{1}{2}} = 1 - \sqrt{2 \times \frac{1}{2} - \frac{1}{2}} = 1 - \sqrt{0.5} = 0.29289$. In our RSA variant, the private exponent is of 568 bits. Therefore, $\delta = \frac{568}{1024} = 0.5546875$ which is out of the range of $0 < \delta < 0.29289$. So our RSA variant is secure against Weger's attack [20].

**Defending Against the Cubic Attack.** According to Section 2.3.3 for the cubic attack, in our variant, $k$ is of 112 bits, $p$ is of 512 bits, and $e$ is of 568 bits. It is clear that $\log_2 k + \log_2 p \approx 112 + 512 = 624 >> \frac{1}{3}\log_2 e \approx \frac{1}{3} \times 568$, In such a case, the cubic attack cannot work.

**Defending Against an Exhaustive Search.** One can check a guess for $k$ since $\phi(N) = N + 1 - (p+q) \equiv (-k)^{-1} \pmod{e}$ and so $(p+q) \equiv N + 1 + k^{-1} \pmod{e}$. Since $p + q < e$, this gives $p + q$ exactly and then we can test the guess by checking whether $a^{N+1-(p+q)} \equiv 1 (\bmod\ N)$ for a random value $a$. In our proposed scheme, $k$ is large enough (112 bits), an exhaustive search method can not work effectively.

**Defending Against Other Attacks.** We also consider the extensions of the above attacks, including the Verheul and Tilborg attack [19], the Weger attack [20], and Hong *et al.*' attack [7]. There is no evidence showing that the proposed scheme is insecure under these extensions. We also try to construct new polynomial equations in which we expect to solve their roots using Coppersmith's lattice technique. So far we are unable to find any useful polynomial equation to do that. Note that it is still an open problem if there exists any polynomial-time algorithm for breaking Sun et al.'s second RSA variant. This also implies that so far no feasible attacks can work well on the new RSA variant because breaking the new RSA variant would be more difficult than breaking Sun et al.'s second RSA variant.

## 5 New RSA Variant with Balanced Prime Factors and Trade-Off Exponents

Sun et al.'s third RSA variant is designed for rebalancing the computation cost between encryption and decryption. By this method, one may shift the work from decryptor to encryptor due to $\log_2 e + \log_2 d \approx \log_2 N + l_k$, where $l_k$ is a predetermined constant, e.g., $l_k$=112. However, the constructed RSA has the limitation of $\log_2 p + \log_2 d \leq \log_2 N$ (assuming $p < q$). That means that if we make both $p$ and $q$ have the same length, $\frac{1}{2}\log_2 N$, the instances that can be constructed by Sun et al.'s scheme are only those instances whose $d$ are of $\frac{1}{2}\log_2 N$ bits, and $e$ are of $\frac{1}{2}\log_2 N + l_k$ bits. Note that in the past, Sakai et al. [16] proposed a key generation algorithm for RSA which provides the similar goal as Sun et al.'s third variant. Regrettably, their algorithm is insecure due to [17, 18]. In this section, we present a new RSA variant with balanced $p$ and $q$ and $\log_2 e + \log_2 d \approx \log_2 N + l_k$ without any other constraint. Without loss of generality, we assume $d < e$. If $d > e$, we need only interchange them.

**Scheme B:** input: $l_N, l_d$, and $l_k$; output: $e, d, p, q$ and $N$.

Step 1. Randomly select a prime $p$ of $\frac{1}{2}l_N$ bits.

Step 2. Randomly select a number $k^{'}$ such that $k^{'}(p-1)$ is of $l_N + l_k - l_d$ bits, where $l_k$ is a security parameter, e.g., $l_k = 112$, and $l_d$ is the bit-length of $d$.

Step 3. Randomly select a number $d$ of $l_d$ bits, such that $\gcd(k^{'}(p-1), d) = 1$.

Step 4. Determine $u^{'}, v^{'}$ such that $du^{'} - k^{'}(p-1)v^{'} = 1$, where $0 < u^{'} < k^{'}(p-1)$ and $0 < v^{'} < d$.

Step 5. If $l_{v^{'}} < l_d$, then assign $u^{'} = u^{'} + k^{'}(p-1)$ and $v^{'} = v^{'} + d$.

Step 6. (Case I) If $l_d > \frac{1}{2}l_N$, try to find $v^{'} = k^{''}q^{'}$, where $l_{k^{''}} = (l_d - \frac{1}{2}l_N)$ and $q^{'}+1$ is a prime. If it fails, go to Step 3; else $e = u^{'}$, $q = q^{'}+1$, and $N = pq$.
(Case II) If $l_d \leq \frac{1}{2}l_N$, try to find $k^{'} = k^{''}t$, where $l_{k^{''}} = l_k$ and $tv^{'}+1$ is a prime. If it fails, go to Step 3; else $e = u^{'}$, $q = tv^{'}+1$, and $N = pq$.

Here we omit to analyze the feasibility for this algorithm and the security for this variant because these analyses are very similar to those of Scheme A. Instead, we illustrate two instances constructed from this variant. The first instance has $p$ and $q$ of 512 bits, $d$ of 540 bits, and $e$ of 596 bits; and the other one has $p$ and $q$ of 512 bits primes, $d$ of 512 bits, and $e$ of 624 bits. These two examples are shown in Appendix C.

## 6 Implementations for the Proposed Schemes

In order to show that our schemes are actually feasible, we implemented our algorithms and measured the average running time for three different sizes of RSA. The main component in our implementations is the factorization method. In our implementations, we select Pollard $p-1$ method [13] as our fundamental factorization method. Furthermore, the programming language used for our implementations is C under NTL with GMP (GNU Multi-Precision library) on Windows systems using Cygwin tools. The machine we used is a personal computer (PC) with 2.8GHz CPU and 512MB DRAM. We consider three different cases for comparisons. The first case has $p$ and $q$ of 512 bits, $d$ and $e$ of 568 bits; the second case has $p$ and $q$ of 512 bits, $d$ of 540 bits, and $e$ of 596 bits; the third case has $p$ and $q$ of 512 bits, $d$ of 512 bits, and $e$ of 624 bits.

Table 3 shows the results and conditions for generating RSA key pairs in our schemes. The item "B_Bound", a predetermined integer using the Pollard $p-1$ method, denotes the upper bound for all prime power divisors of $p-1$. This value is chosen by experience in our program. The item "AverageTime" denotes the average running time for each case upon testing 100 samples. The item "AverageLoopNum" counts the number of loops running from Step 3 to Step 6. Note that what we are doing in Step 6 of our implementations is only to find small factors of $v^{'}$ and then try to compose part of these small factors into what we need. According to our experiments, if one tries to factor $v^{'}$ completely, then "AverageLoopNum" will be smaller, but "AverageTime" will be longer because

**Table 3.** Experimental results in PC platform of 2.8GHz CPU, 512M DRAM.

| | Scheme A | Scheme B | |
|---|---|---|---|
| Input (Bit-length) | $l_N = 1024$ $l_e = 568$ $l_d = 568$ $w = 56$ | $l_N = 1024$ $l_e = 596$ $l_d = 540$ $l_k = 112$ | $l_N = 1024$ $l_e = 624$ $l_d = 512$ $l_k = 112$ |
| B_Bound | 150 | 30 | |
| AverageTime (sec) | 1060.93 | 20.61 | 0.46 |
| AverageLoopNum | 290490 | 29273 | 319 |

the time will be dominated by the factorization. From Table 3, we know that the more balanced $e$ and $d$ are, the more time-consuming our algorithms are. The most time-consuming case is exactly Scheme A. The average time for generating such a key pair is about 16 minutes under our implementations. This may be heavy for the end user's use. However, it can be much improved by some parallel techniques and/or high-end computers in the case when the RSA key pair must be generated and issued by centralized control. For example, a trusted CA issues smart cards in which every user's private key, public key, and the corresponding certificate are embedded by a smart card writer.

## 7 Discussion and Application

Comparing with the typical RSA with small $e$ and randomly determined $d$, Scheme A is about twice faster in decryption, but the public exponent $e$ is about of $\frac{1}{2}l_N$ bits. On the other hand, RSA-CRT achieves 4 times faster and can choose small $e$, e.g. $e$=$2^{16}$+1. Thus, our variants can not provide better performance than RSA-CRT. However RSA-CRT needs to keep more secrets ($d_p$, $d_q$, $p$, and $q$) than our schemes. Besides, RSA-CRT usually brings on some additional security problems [9]. In the following, we further propose an application, based on RSA, to entity authentication for defending a type of attack, called the stolen-secret attack. It is remarked that our RSA variants can be applied to realize such an application, while RSA-CRT can not.

With two-party authentication protocols in place, it would be easy for one participant to establish trusted communication with the other. In general, there are three approaches to designing authentication protocols. The first approach is based on the public-key cryptosystem (involving signature mechanism). This approach works under PKI environment and needs a trusted CA to support. The second approach is based on a shared password which is easy to remember by user. This approach usually need to be designed to defend the dictionary attack. The third approach is based on a shared secret-key of a symmetric cryptosystem. This approach uses symmetric-key encryption to validate the identity of protocol participants. Here we consider the *stolen-secret attack* in which an adversary who has stolen the secret (a private key, or a shared password, or a shared secret-key) from one party can use it directly to masquerade as the other party. Among these three approaches, the first approach is secure against the stolen-secret attack because one party's private key leaked will not lead to a forgery of the other party. However, it is only suitable for the environment with CA and PKI supporting. For the password-based protocol, because two parties share a common password, therefore it is insecure against the stolen-secret attack. An improvement for this approach is called the verifier-based protocol in which one party (client) keeps a password and the other one (server) keeps the corresponding verifier (usually it is a hashed image of the password). Thus if the verifier is leaked, it will not lead a forgery of the client. However, if the password is leaked (on the client side), this will lead to a successful forgery of the server because the verifier can be easily computed from password. As for the third approach, it is clear that the stolen-secret attack can work well.

As mentioned above, the stolen-secret attack is a baffling problem in authentication protocols. Here we attempt to enhance the secret-key based protocol to defend the stolen-secret attack. In general, RSA system generates a key pair ($e$,$d$), where the public key $e$ is disclosed and the private key $d$ is disguised. If both $e$ and $d$ are kept secret by two parties respectively, and $p$ and $q$ are unknown to any one. Thus we can regard RSA as a secret-key cryptosystem. We imagine that a key distribution center generates an RSA key pair ($e$,$d$) by using the key generating algorithm in Scheme A. And then $e$ is kept secret by Alice and $d$ is kept secret by Bob. The RSA modulus $N$ is public but no one knows $p$ and $q$ exactly. Thus, neither Alice nor Bob can obtain the secret of each other. Note that it is clear that RSA-CRT can not be used in such a situation because $p$ and $q$ are unknown by any party.

In 1993, Bellare and Rogaway [1] proposed two provably secure symmetric-key authentication protocols, MAP1 and MAP2. MAP1 is a mutual authentication protocol for two parties, and MAP2 allows arbitrary text strings to be authenticated along with its flows. As our examples of defending the stolen-secret attack, we modify MAP1 and MAP2 in the following. A brief outline of these two protocols and our improvements are presented in Fig. 1 and Fig. 2. Here $A^a$ and $B^a$ denote that Alice keeps a shared secret key $a$ with Bob; and $A^e$ and $B^d$ denote that Alice keeps $e$ and Bob keeps $d$, where ($e$,$d$) is a key pair of RSA using our scheme A. Let $R_X$ denote a random challenge from $X$ and $[x]_k = (x, f_k(x))$,where $f_k(x)$ is a pseudorandom function family specified by key $k$. It is commonly believed that pseudorandom functions can be well-implemented by encryption primitives in practice. Here we replace $f_k(x)$ by either a symmetric-key encryption with key $k$ (in MAP1 and MAP2) or an encryption of RSA with exponent $k$ (in the improved MAP1 and the improved MAP2). Here we remark that the plain RSA encryption can not be used directly for practical purpose, some padding techniques, such as PKCS #1 and OAEP, are required. We also note that although we limit our discussion to authentication protocols, there exists the even more important concept of key-distribution, often coupled with authentication. Our improvements to defend the stolen-secret attack can be also applied to those key distribution protocols whose security are based on symmetric-key encryption.

$$\begin{array}{lcr} A^a & & B^a \\ & \xrightarrow{R_A} & \\ & \xleftarrow{[B.A.R_A.R_B]_a} & \\ & \xrightarrow{[A.R_B]_a} & \end{array} \quad \Longrightarrow \quad \begin{array}{lcr} A^e & & B^d \\ & \xrightarrow{R_A} & \\ & \xleftarrow{[B.A.R_A.R_B]_d} & \\ & \xrightarrow{[A.R_B]_e} & \end{array}$$

**Fig. 1.** MAP1 and Improved MAP1.

$$\begin{array}{lcr} A^a & & B^a \\ & \xrightarrow{R_A.Text_1} & \\ & \xleftarrow{[B.A.R_A.R_B.Text_1.Text_2]_a} & \\ & \xrightarrow{[A.R_B.Text_3]_a} & \end{array} \quad \Longrightarrow \quad \begin{array}{lcr} A^e & & B^d \\ & \xrightarrow{R_A.Text_1} & \\ & \xleftarrow{[B.A.R_A.R_B.Text_1.Text_2]_d} & \\ & \xrightarrow{[A.R_B.Text_3]_e} & \end{array}$$

**Fig. 2.** MAP2 and Improved MAP2.

## 8 Conclusions

As shown by Durfee and Nguyen, the more unbalanced the prime factors are, the more insecure Sun et al.'s RSA variants are. In this paper, we propose a new RSA variant with balanced prime factors and balanced exponents. It is clear that this proposed variant is more secure than Sun et al.'s second RSA variant with unbalanced prime factors and balanced exponents. As an example, we can construct an instance of RSA with $p$ of 512 bits, $q$ of 513 bits, and $d$ and $e$ of 568 bits. In addition, for repairing the security of Sun, Yang, and Laih's third RSA variant, we also present another RSA variant with balanced prime factors and $\log_2 e + \log_2 d \approx \log_2 N + l_k$ . This variant is designed for rebalancing the computation cost between encryption and decryption. It should be noted that in this variant the private exponent $d$ must be large enough to defend against the Durfee-Nguyen attack and its extensions. Based on RSA, we also give an application to entity authentication in order to defend the stolen-secret attack. Our RSA variants can be applied to realize such an application, while RSA-CRT can not.

## Acknowledgements

The authors wish to acknowledge the anonymous reviewers for valuable comments and thank helpful discussions with Chiung-Hsun Chen, Mu-En Wu and Ting-Yao Lin on several points in the paper. This research was supported in part by the National Science Council, Taiwan, under contract NSC-93-2213-E-007-102.

## References

1. Bellare,M., Rogaway, P.: Entity authentication and key distribution. Proceedings of Crypto '93 , LNCS 773 (1994) 232-249
2. Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key $d$ less than $N^{0.292}$. Proceedings of Eurocrypt '99, LNCS 1592 (1999) 1–11

3. Cavallar, S., Dodson B., Lenstra, A. K., Lioen, W., Montgomery, P. L., Murphy, B., te Riele, H., Aardal, K., Gilchrist, J., Guillerm, G., Leyland, P., Marchand, J., Morain, F., Muffett, A., Putnam, C., Putnam, C., Zimmermann,P.: Factorization of 512-bit RSA key using the number field sieve. Proceedings of Eurocrypt'00, LNCS 1807 (2000) 1-18
4. Coppersmith, D.: Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. Proceedings of Eurocrypt'96, LNCS 1070 (1996) 178–189
5. Durfee, G., Nguyen, P.: Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt'99. Proceedings of Asiacrypt'00, LNCS 1976 (2000) 14–29
6. Herstein, I. N.: Topics in Algebra, Xerox Corporation. (1975)
7. Hong, H. S., Lee, H. K., Lee, H. S., Lee, H. J.: The better bound of private key in RSA with unbalanced primes. Applied Mathematics and Computation, Vol. 139 (2003) 351-362
8. http://www.alpertron.com.ar/ECM.HTM
9. Joye,M., Quisquater, J. J., Yen, S. M., Yung, M.: Security paradoxes: how improving a cryptosystem may weaken it. Proceedings of the Ninth National Conference on Information Security (1999) 27-32
10. Lenstra,A., Lenstra, H., Lovasz, L.: Factoring polynomial with rational coefficients. Mathematiche Annalen, Vol. 261 (1982) 515-534
11. Lenstra Jr, H. W.: Factoring integers with elliptic curves. Annuals of Mathematics, vol. 126 (1987) 649–673
12. Pinch, R.: Extending the Wiener attack to RSA-type cryptosystems. Electronics Letters, Vol. 31 (1995) 1736-1738
13. Pollard, J.: Theorems of factorization and primality testing. Proc. Cambridge Philos. Soc., (1974) 76:521–528
14. Rivest, R. L., Shamir, A., Adleman, L. M.: A method for obtaining digital signatures and public-key cryptosystems. Comm. ACM, Vol. 21 (1987) 120-126
15. Rivest, R., Silverman, R. D.: Are strong primes needed for RSA?. The 1997 RSA Laboratories Seminar series, Seminar Proceedings (1997)
16. Sakai, R., Morii, M., Kasahara, M.: New key generation algorithm for RSA cryptosystem. IEICE Transactions on Fundamentals, Vol. E77-A (1994) 89-97
17. Sun, H. M., Yang, W. C., Laih, C. S.: On the design of RSA with short secret exponent. Proceedings of Asiacrypt'99, LNCS 1716 (1999) 150–164
18. Sun, H. M., Yang, W. C., Laih, C. S.: On the design of RSA with short secret exponent. Journal of Inforamtion Science and Engineering, Vol.18 No.1 (January 2002) 1-18
19. Verheul, E., van Tilborg, H.: Cryptanalysis of less short RSA secret exponents. Applicable Algebra in Engineering, Communication and Computing, Vol. 8 (1997) 425-435
20. de Weger, B.: Cryptanalysis of RSA with small prime difference. Applicable Algebra in Engineering, Communication and Computing, Vol. 13 (2002) 17-28
21. Wiener, M.: Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory, Vol. 36, no. 3 (1990) 553–558

## Appendix A: Sun, Yang, and Laih's RSA Variants

*Scheme(I).* input: $l_N, l_p, l_d$, and $\gamma$; output: $e, d, p, q$ and $N$.

Step 1. Select two random primes $p < q$ such that both $p$ and $N$ are sufficiently large to defend factorization algorithms such as ECM [11] and NFS [3].

Step 2. Randomly select the secret exponent $d$ such that $l_d + l_p > \frac{1}{3}l_N$ and $d > 2^{\gamma}p^{0.5}$, where $\gamma$ is the security parameter (larger than 64).
Step 3. If the public exponent $e$ defined by $ed \equiv 1(\mathrm{mod}\phi(N))$ is not larger than $\frac{\phi(N)}{2}$, one restarts the previous step.

It is clear that for the first RSA variant, the improved one with balanced $p$ and $q$ is, in fact, the standard RSA. Hence, it is impossible to make $d$ short below Boneh and Durfee's bound and Wiener's bound.

*Scheme(II).* input: $l_N$; output: $e, d, p, q$ and $N$.

Step 1. Randomly select a prime $p$ of $\frac{1}{2}l_N - 112$ bits, and a $k$ of 112 bits.
Step 2. Randomly select a $d$ of $\frac{1}{2}l_N + 56$ bits coprime with $k(p-1)$.
Step 3. We can unique determined two numbers $u$ and $v$, such that $du - k(p-1)v = 1$,where $0 < u < k(p-1)$ ,$0 < v < d$.
Step 4. If $\gcd(v+1, d) \neq 1$,then go to step 2.
Step 5. Select a number $h$ of 56 bits until $q = v + hd + 1$ is prime.
Step 6. Let $p$,$q$,$e = u + hk(p-1)$,$d$, and $N = pq$ are the parameters of RSA.

For the second RSA variant, it is impossible to make $p$ and $q$ balanced because $p$ is of $\frac{1}{2}\log_2 N - 112$ bits and $q$ is of $\frac{1}{2}\log_2 N + 112$ bits in this variant.

*Scheme(III).* input: $l_N, l_p, l_d$,and $l_k$; output: $e, d, p, q$ and $N$.

Step 1. Randomly select a prime number $p$ of length $l_p$, such that it is large enough to make an ECM [11] attack infeasible.
Step 2. Randomly select a number $k$ of length $l_k$.
Step 3. Randomly select a number $d$ of length $l_d$ and $\gcd(k(p-1), d) = 1$.
Step 4. we can uniquely determine two numbers $u^{'}$ and $v^{'}$ such that $du^{'} - k(p-1)v^{'} = 1$, where $0 < u^{'} < k(p-1)$ and $0 < v^{'} < d$.
Step 5. If $\gcd(v^{'} + 1, d) \neq 1$, then go to Step 3.
Step 6. Randomly select a number $h$ of length $l_N - l_p - l_d$ , then compute $u = u^{'} + hk(p-1)$ and $v = v^{'} + hd$.
Step 7. If $v + 1$ is not a prime number, go to Step 6.
Step 8. Let $p$, $q = v + 1$, $e = u$, $d$, and $N = pq$ are the parameters of RSA.

For the third RSA variant, the possibly constructed RSA with balanced $p$ and $q$ are only those instances of RSA with $d$ of $\frac{1}{2}\log_2 N$ bits and $e$ of $\frac{1}{2}\log_2 N + l_k$ bits, e.g., $l_k$=112. This is due to the limitation of $\log_2 p + \log_2 d \leq \log_2 N$.

## Appendix B: An Instance of RSA with Balanced Prime Factors and Balanced Exponents

As an example for Scheme A, we construct an instance of RSA with $p$ of 512 bits, $q$ of 513 bits, $d$ and $e$ of 568 bits.

$p =$ EB73E838 FE3A755B 1B08C0A5 4070CF38 62046A3D 77E26D54 73EB8541 6662E060 25388EC1 17129F9F D3F7E81A 81CC11DC 0ED30F96 39E201C4 FAC77E73 73B75CDD

$q =$ 1 E47C6F97 82515CEE 69DA0782 A1D1DEF3 A7F15B88 F513242F
CF505867 24AB9F4F 39349987 006B5AE6 3A0FBFA7 A7BBFAC7 8D6B0BEE
04089C0C 7F82C605 85A66B79

$d =$ F34255 6EB55834 5EB2023d 33DA5792 8C373385 86B72B71 D0A19BB6
4B490155 74BBB648 287F297F 865313B7 4F17982D D854F694 82C19436
91F7FB5B B73BE6CB 66952AC4 1A416E69

$e =$ D01CD7 7DA75CA6 39247A84 45E39813 B98BF2DC 13DEEC98 D31725A4
52F83345 0647E852 0CA70032 600B582B 1B2BB83F 9DF38D6E 1F73069C
C2B05BCB 81710127 D33D9414 D5654D39

## Appendix C: Two Instances of RSA with Balanced Prime Factors and Trade-off Exponents

As an example for Scheme B, we construct an instance of RSA with $p$ and $q$ of 512 bits, $d$ of 540 bits, and $e$ of 596 bits.

$p =$ DE7332C0 6DDB34F5 86598C8F 2F103983 EE86007F DFB44CBF F503F1EB
F4BCD507 23EA54EA 5E9AE43F 7FC54021 CD026D8B C23B48CD D00ECDA2
9054EBB5 C5A6D063

$q =$ 89575BE4 F0310066 113CF04C 1220DAB7 25DD3F2F DD59BA09 3CC31FAC
467D17F9 2FA38A26 72D92E32 B91333FA 88F1D013 E5EB1A74 E4DE793E
E9A299A9 A7C0D24B

$e =$ 88E33 2BF9879D 6AD5324B 6763FB22 E6D21B8D CB28E5E5 437AA101
D27D7992 42E507D3 D2639902 C58C4978 D79D5A0A CF515FA0 028662AF
5F26F0FB AF60DF38 8E4409F3 63AE6806 B2045771

$d =$ 8BB9953 6F0577AC DF1D6DB8 0F76A4CF 992F8538 FC89BEB6 5DEA50E1
124AB868 9BD989B3 D20A8EC9 B3D697AF 76F1C16F 4BD09BBC C8E53CCB
AC16B232 FD39134E 7E913009

As another example for Scheme B, we construct an instance of RSA with $p$ and $q$ of 512 bits primes, $d$ of 512 bits, and $e$ of 624 bits.

$p =$ 84A0CC27 66ACCDA9 57646FC5 924AA056 5E2AC1DA 1137B9DB AC6BE9D2
DD09FA82 193D6205 0E62C4BD 0D2A0304 037DED34 03290E3A 748C6AF4
80FB6880 828CF3A3

$q =$ B187BA5F AB9CABEC 765897BA B364DB52 D8959D5C B765A725 1A1EDCA3
19F9601D 2CE5D8A9 570386BB 1F016B40 6DDBE6C2 EBEA445F 14D48FD4
B7177E03 F4959BFF

$e =$ D260 A347D9C1 76B8BC8B DB527877 F09489C0 E634E313 4A7FAB5C
A135EB1D A6410CBC CD497FB7 092C3CB2 2BA23E7D D02201B3 ABD9E989
584ED3C7 262A3ED0 CEFD6757 00E7B6DC 414D77BA 050BF525

$d =$ 91273082 5084AB61 D38E2142 3AED897E 97FBDCEC 00081122 3FCF3B70
E3D5D8BE A5AD07F5 B0D67990 6C253F89 30A26574 F80CD0F6 A007AE0A
6C131816 E85A4B35